Chapter 14: GUI Development and Application Design in SciLab

Introduction

Graphical User Interface (GUI) development plays a vital role in modern scientific and engineering applications, enabling users to interact with software tools more intuitively. SciLab, an open-source platform for numerical computation, provides powerful tools for GUI creation through its dedicated module called **GUI Builder** and **uicontrols**. This chapter explores the techniques of designing and implementing interactive applications using SciLab's GUI capabilities. We will walk through the development process, from basic GUI elements to full-scale application design, all within SciLab's environment.

14.1 GUI Builder in SciLab

14.1.1 Introduction to GUI Builder

- GUI Builder is a toolbox in SciLab that provides a drag-and-drop interface for creating user interfaces.
- It simplifies the process of placing components like buttons, sliders, frames, etc.
- The generated GUI code can be customized and embedded into larger applications.

14.1.2 Installing the GUI Builder Module

- Accessing ATOMS (Autonomous Modules)
- Using the following command to install:
- `atomsInstall("GUI Builder")`
- Loading the toolbox after installation:
- `atomsLoad("GUI Builder")`

14.1.3 Launching GUI Builder

- After loading, launch using:
- `guibuilder()`
- Introduction to the GUI Builder interface: toolbar, canvas, properties pane.

14.2 Basic GUI Components

14.2.1 Push Button

- Creation using GUI Builder or programmatically:
- `uicontrol("style","pushbutton","string","Click Me","position",[100 100 100 40])`
- Callback definition using `"callback"` property.

14.2.2 Static Text and Editable Text

- Static text:
- `uicontrol("style","text","string","Display Here","position",[80 160 100 20])`
- Editable text box for input:
- `uicontrol("style","edit","position",[80 190 100 30])`

14.2.3 Checkboxes and Radio Buttons

- Checkbox:
- `uicontrol("style","checkbox","string","Option A","position",[50 250 100 30])`
- Radio button group creation with appropriate grouping logic.

14.2.4 Sliders and List Boxes

- Slider:
- `uicontrol("style","slider","position",[50 300 150 20],"min",0,"max",100,"value",50)`
- List box with multiple entries:
- `uicontrol("style","listbox","string",["Option1","Option2","Option3"],"position",[220 100 100 80])`

14.3 Event Handling and Callback Functions

14.3.1 What are Callbacks?

- Callbacks are scripts executed when a user interacts with a component (e.g., clicking a button).
- Defined using the `"callback"` property of GUI elements.

14.3.2 Writing Callback Functions

- Defining functions in the same file or as separate `.sci` scripts.
- Using `global` variables to share data between callbacks.
- Example:
- ```
  function onButtonClick()
  disp("Button clicked!")
  endfunction
  ```

14.3.3 Linking GUI Components with Callbacks

- Assigning the function:
- `uicontrol("style","pushbutton","string","Click","callback","onButtonClick()")`

14.4 Layout Management and GUI Design Principles

14.4.1 Positioning and Alignment

- Position vector `[x, y, width, height]`
- Absolute positioning vs dynamic layout.
- Using grids or relative placement for scalability.

14.4.2 Frames and Panels

- Using frames to group controls:
- `uicontrol("style","frame","position",[50 50 200 150])`

14.4.3 Best Practices in GUI Design

- Keeping the interface intuitive.
- Using meaningful labels and tooltips.
- Avoiding clutter; organizing components logically.

14.5 Advanced GUI Applications

14.5.1 Passing Data Between Components

- Use of `global` or persistent variables.
- Reading and updating component properties dynamically using:
- ```
  h = uicontrol(...);
  h.value = 50;
  ```

14.5.2 Real-time Data Input and Output

- Linking GUI with functions performing computations.
- Updating GUI elements with results (e.g., displaying in static text).

14.5.3 File I/O via GUI

- Letting users load and save files through GUI.
- Using `uigetfile()` and `uiputfile()` functions.

14.5.4 GUI with Plotting Capabilities

- Integrating `plot2d` within GUI callbacks.
- Example:
- ```
  function plotGraph()
  x = 0:0.1:10;
  y = sin(x);
  clf();
  plot2d(x, y)
  endfunction
  ```

14.6 Packaging and Deployment of GUI Applications

14.6.1 Creating Executable SciLab GUI Scripts

- Consolidating GUI code and logic.
- Saving the file as `.sce` or `.sci`.

14.6.2 Distributing Applications

- Creating a script that loads the GUI on SciLab start.
- Bundling dependencies (like toolboxes).

14.7 Debugging and Testing GUI Applications

14.7.1 Debugging Tools in SciLab

- Using `disp()` and `mprintf()` for debug outputs.
- Step-by-step testing of callbacks.

14.7.2 Common GUI Development Errors

- Misaligned positions
- Unresponsive callbacks
- Errors in reading component values (e.g., wrong handle)
  ```

14.7.3 Performance Optimization Tips

- Avoid unnecessary redraws or computations.
- Use appropriate data structures for storing component references.

14.8 Case Study: Building a Simple Calculator in SciLab GUI

14.8.1 Objective

- Create a GUI-based calculator for basic arithmetic operations.

14.8.2 Interface Layout

- Two text input fields.
- Four buttons for `+`, `-`, `*`, `/`.
- One static text for displaying the result.

14.8.3 Implementing Functionalities

- Reading input values from edit boxes.
- Performing operations based on button click.
- Displaying output in result field.

14.8.4 Code Walkthrough

- Complete functional code with proper callback assignment.
